



FA2023 Week 04 • 2023-09-21

Reverse Engineering I

Richard and Juniper

Announcements

2023-09-23 • This Saturday!

- **Fall CTF 2023**
- First 350 registered people to show up (sigpwny.com/register23) get an electronic badge!
Also, free shirt + pizza!



Scoreboard

Place	User	Score
1	ronanboyarski	23485
2	NullPoExc	21615
3	caasher	13860
4	EhWhoAml	8195
5	CBCicada	7260
6	mgcsstywth	6475
7	SHAD0VV1RUS	5970
8	aaronthewinner	5555
9	drizzle	4825
10	Ex-32	4230



ctf.sigpwny.com

sigpwny{r3v3r5e_eng1n3er1ng}



Which is easier to understand?

```
def fibonacci(n):  
    if n <= 0:  
        return []  
    elif n == 1:  
        return [0]  
    elif n == 2:  
        return [0, 1]  
    else:  
        fib_sequence = [0, 1]  
        while len(fib_sequence) < n:  
            next_num = fib_sequence[-1] + fib_sequence[-2]  
            fib_sequence.append(next_num)  
        return fib_sequence
```

```
def aaoaaaa04922(aa27619):  
    aaoaaaa20551 = -1  
    aa27619 = aa27619 + 1  
    aa27618 = -aa27619  
    if aa27618 > 0:  
        return []  
    elif not bool(aa27619 - 2):  
        return [] * aa27618  
    elif aa27619 == 1:  
        return [aa27619-1]  
    else:  
        aaoaaaa32021 = [0, 1]  
        while True:  
            if not (len(aoaaaa32021) < aa27619):  
                break  
            aaoaaaa21049 = aaoaaaa32021[-aaoaaaa32021[1]]  
            aaoaaaa21049 += aaoaaaa32021[aaoaaaa20551**2 - 3]  
            aaoaaaa32021.append(aaoaaaa21049)  
        else:  
            aaoaaa3322 = 23  
            return [aaoaaa3322 + i for i in aaoaaaa32021]  
    return aaoaaaa32021
```

Overview

- Basics
 - Motivation
 - Types of analysis
 - Abstraction levels
- Techniques
 - Known algorithms
 - Tools
- Examples



Basics

What is reverse engineering?



Motivation

- Reverse engineering: reason about original meaning of code
- Goal is to understand the code
 - The code is never "wrong" — it is the ultimate "documentation"
- Not all code is easy to read or well-documented
- Sometimes code is intentionally hard to understand



Static vs Dynamic Analysis

- Static Analysis
 - Reading code
 - Using tools to understand code

- Dynamic Analysis
 - Running code
 - Inspecting program state as it is running

Most helpful if...

- Code is simple
- Code is hard to run

Most helpful if...

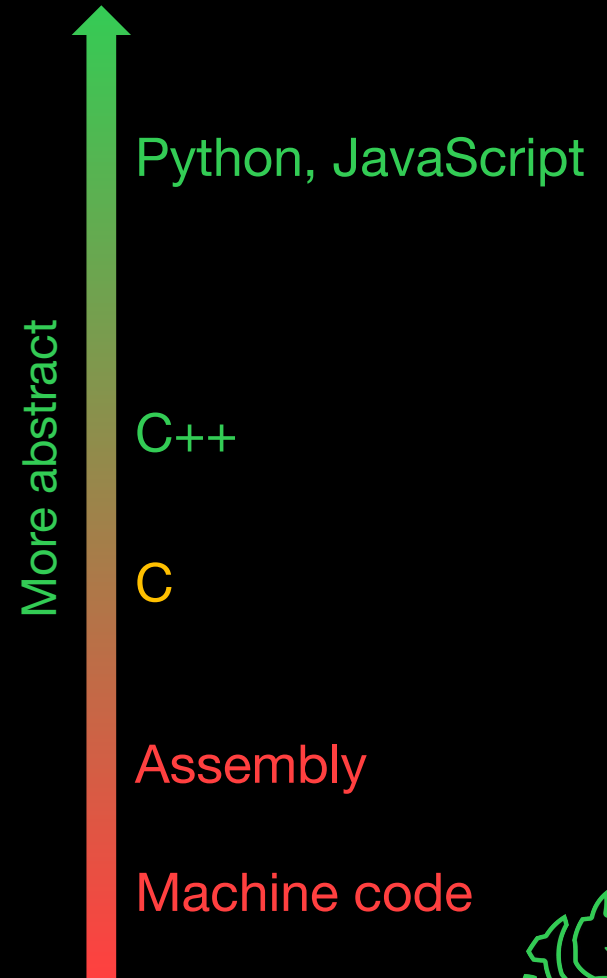
- Code is complex

Static and dynamic analysis are not a dichotomy! Use them together!



Abstraction Levels

- High level
 - Python, JavaScript, etc.
 - Easy to analyze
- Low level
 - C, assembly, etc.
 - Hard to analyze
 - Everything is ran as machine code at some point



Example: Dynamic analysis



Give me a number: 4

4

Give me a number: 10

11

Give me a number: 0

2

Give me a number: -4

-1

Give me a number: 7

What is the next output?

What does the code look like?



The Code



```
i = 0
while True:
    val = int(input('Give me a number: '))
    i += 1
    print(val + i)
```



Making Assumptions

- Occam's razor: the simplest solution is often the right one
- But always remember that assumptions can be wrong



```
i = 0
while True:
    val = int(input('Give me a number: '))
    i += 1
    if i > 20:
        val = -val
    print(val + i)
```

What if the previous code looked like this?



Techniques

How to reverse engineer?



Static Analysis

- Function rewriting
 - Simplify complex portions of code
- Find known algorithms/patterns
- Decompilers
 - Automatically extract abstractions from low level programs
 - Turn assembly into more readable C
 - Will be covered in depth in Reverse Engineering II meeting



Example: Common patterns



```
arr = [0] * 10
i = 9
while i >= 0:
    arr[i] = i * 2
    i -= 1
```

Can you simplify this code?



Example: Common patterns



```
arr = [0] * 10  
for i in range(9, 0 - 1, -1):  
    arr[i] = i * 2
```

Simplify even more?



Example: Common patterns



```
arr = [0] * 10
for i in range(0, 10):
    arr[i] = i * 2
```

Even simpler?



Example: Common patterns



```
arr = [i*2 for i in range(0, 10)]
```



Example: Common patterns

Which level of simplification is the most useful?



```
arr = [0] * 10
i = 9
while i >= 0:
    arr[i] = i * 2
    i -= 1
```



```
arr = [0] * 10
for i in range(9, 0 - 1, -1):
    arr[i] = i * 2
```



```
arr = [0] * 10
for i in range(0, 10):
    arr[i] = i * 2
```



```
arr = [i*2 for i in range(0, 10)]
```



Dynamic Analysis

- **Partial evaluation**
 - Evaluate small portions of the code to reduce complexity
- **Modifying programs**
 - Add or remove code
 - Add print statements
 - "Patching" binaries



Advanced Dynamic Analysis

- Debuggers
 - gdb, pdb
 - Will be covered in depth in Reverse Engineering II meeting
- Side channels
 - Instruction counting, time counting
 - Will be covered in depth in Reverse Engineering II meeting



Examples



Goal: input flag to pass checks
(doesn't print "wrong")

```
f = input('What is the flag? ')
va = [1751, 1649, 1836, 1734]
arr = ''.join([chr(va[len(va)-1-i]//17) for i in range(len(va))])

if f[0] != 'f':
    print("That's definitely wrong.")
else:
    it = len(arr)
    while it > 0:
        it -= 1
        if arr[it] != f[it]:
            print('Wrong flag!')
            exit(1)
```

User input

Rewriting: this is a
backwards for
loop

Don't want this to run





```
f = input('What is the flag? ')

va = [1751, 1649, 1836, 1734]
arr = ''.join([chr(va[len(va)-1-i]//17) for i in range(len(va))])

if f[0] != 'f':
    print("That's definitely wrong.")
else:
    for it in range(len(arr)-1, -1, -1):
        if arr[it] != f[it]:
            print('Wrong flag!')
            exit(1)
```

Two ways to solve: partial evaluation or patching



```
f = input('What is the flag? ')

va = [1751, 1649, 1836, 1734]
arr = ''.join([chr(va[len(va)-1-i]//17) for i in range(len(va))])

print(arr)

if f[0] != 'f':
    print("That's definitely wrong.")
else:
    for it in range(len(arr)-1, -1, -1):
        if arr[it] != f[it]:
            print('Wrong flag!')
            exit(1)
```

One way: patching
(adding print)

```
$ python3 test.py
What is the flag? aaaa
flag
That's definitely wrong.
```



```
$ python3
Python 3.11.5 (main, Aug 24 2023, 15:09:45) [Clang 14.0.3 (clang-
1403.0.22.14.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> va = [1751, 1649, 1836, 1734]
>>> arr = ''.join([chr(va[len(va)-1-i]//17) for i in range(len(va))])
>>> arr
'flag'
```

Second way:
evaluate the
portion in Python
REPL



Go try challenges!

- Go to ctf.sigpwny.com
- Start with **Python RE 1: Easy rev**

- If you don't have Python installed, see slides from last meeting (Reverse Engineering Setup)



Next Meetings

2023-09-23 • This Saturday!

- **Fall CTF 2023**
- First 350 registered people to show up (sigpwny.com/register23) get an electronic badge!
Also, free shirt + pizza!

No meeting this Sunday

2023-09-28 • Next Thursday: OSINT

- ~~Professional stalking~~ Gathering info about people/organizations from freely available sources



ctf.sigpwny.com

sigpwny{r3v3r5e_eng1n3er1ng}

Meeting content can be found at
sigpwny.com/meetings.

